Università degli Studi di Milano

Laurea Specialistica in Genomica Funzionale e Bioinformatica

Corso di Linguaggi di Programmazione per la Bioinformatica

L' ambiente grafico di R

Giorgio Valentini
e –mail: *valentini@dsi.unimi.it*

DSI – Dipartimento di Scienze dell' Informazione
Università degli Studi di Milano

1

Rappresentazioni grafiche in R

- Il linguaggio R è dotato di un ambiente grafico potente e versatile
- E' semplice produrre grafici per l' analisi esplorativa dei dati
- Si possono facilmente generare grafici di elevata qualità utilizzabili per pubblicazioni
- L' ampiente grafico di R può generare grafici in diversi formati (sono disponibili diversi *device driver*):
  - Display diretto su schermo (Linux, Windows e Macintosh)
  - postscript
  - pdf (Adobe Portable Document Format)
  - jpeg (JPEG bitmap)
  - png (PNG bitmap, simile a GIF)
  - wmf (Windows Metafile)

2

Tre gruppi di comandi grafici

1. Funzioni di alto livello:
   creano un nuovo grafico sul device grafico
2. Funzioni di basso livello:
   aggiungono altre parti ad un grafico esistente (ad es: nuove linee, punti o oggetti grafici)
3. Funzioni interattive:
   consentono di aggiungere o estrarre interattivamente informazioni grafiche da un grafico esistente.

Per un esempio di funzioni grafiche in R si esegua:

```
> demo(graphics)
```

3

Comandi di alto livello

plot	La funzione più utilizzata: permette di generare diverse tipologie di grafici (per punti, linee, grafici a barre, etc)
qqnorm, qqline, qqplot	Grafici per confrontare diverse distribuzioni
hist	Generazione di istogrammi
barplot	Grafici a "colonne"
boxplot	Box-and-whisker plot
pie	Grafici a "torta"
image, contour, persp	Comandi per la grafica 3D

Esistono molti altri comandi di alto livello: si veda ad es: dal menu Help di R: help html/packages/graphics

4

Plot

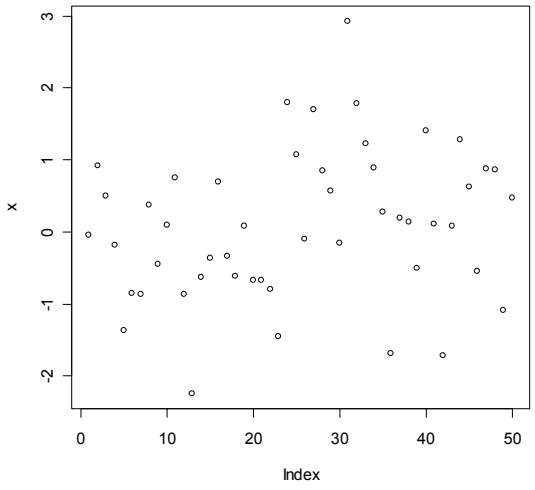
E' una funzione generica di R: il tipo di grafico generato dipende dal tipo o classe del suo argomento:
Esempi:

- plot(x,y) : se x e y sono vettori produce uno scatterplot di x verso y
- plot(X): se X è una matrice a due colonne produce uno scatterplot di una colonna rispetto all' altra
- plot(x): se x è un vettore produce un grafico dei valori del vettore rispetto agli indici
- plot(f): se f è un fattore viene prodotto un barplot
- plot(f,y): se f è un fattore ed y un vettore numerico, viene prodotto un boxplot di y per ogni livello di f
- plot (df): se df è un dataframe, produce i grafici delle distribuzioni delle variabili contenute nel data frame.

Plot dispone di diversi argomenti (si veda l'help).

5

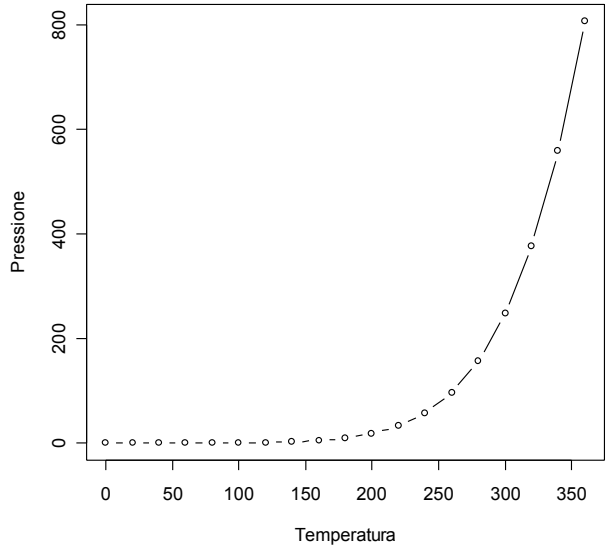
Plot di un vettore numerico




```
> x <- rnorm(50)
> plot(x)
```

6

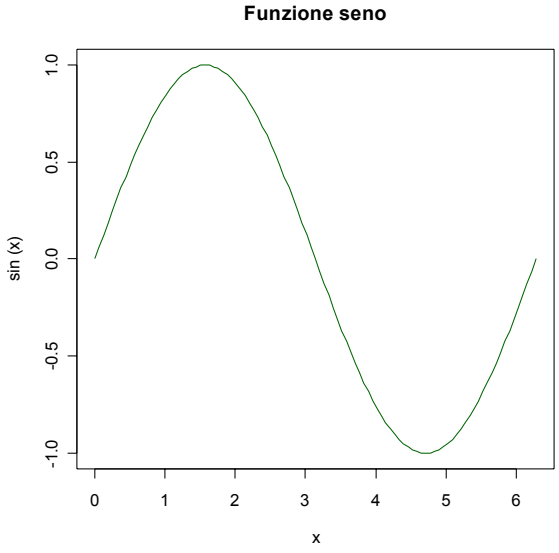
Plot di un vettore numerico rispetto ad un altro




```
> plot(temperature, pressure,
+ xlab="Temperatura", ylab="Pressione", type="b")
```

7

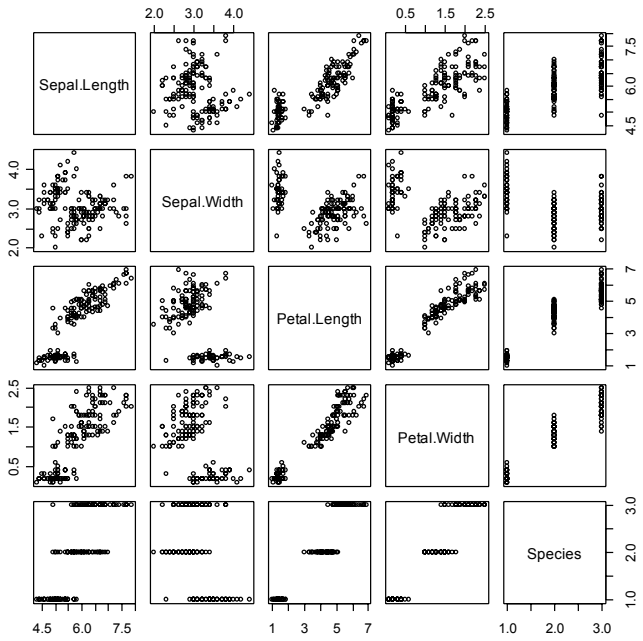
Plot di una funzione




```
> plot(sin, 0, 2*pi, type="l", col="darkgreen", main="Funzione seno")
```

8

Plot di un dataframe




```
> plot(iris)
```

(iris è un dataframe)

9

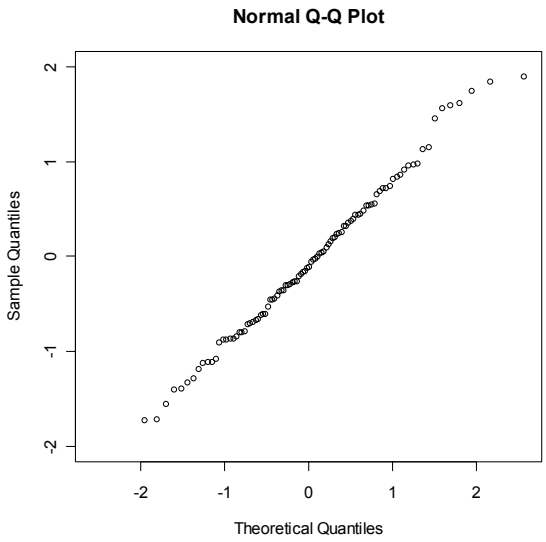
qqnorm, qqline e qqplot

Funzioni per confrontare graficamente diverse distribuzioni:

- qqnorm(x):
  produce un grafico quantile-quantile dei dati del vettore x rispetto ad una corrispondente distribuzione normale
- qqline(x):
  come qqnorm, ma viene aggiunta una linea che passa attraverso il primo e terzo quartile
- qqplot(x,y):
  produce il grafico dei quantili dei dati del vettore x rispetto ai quantili dei dati del vettore y

10

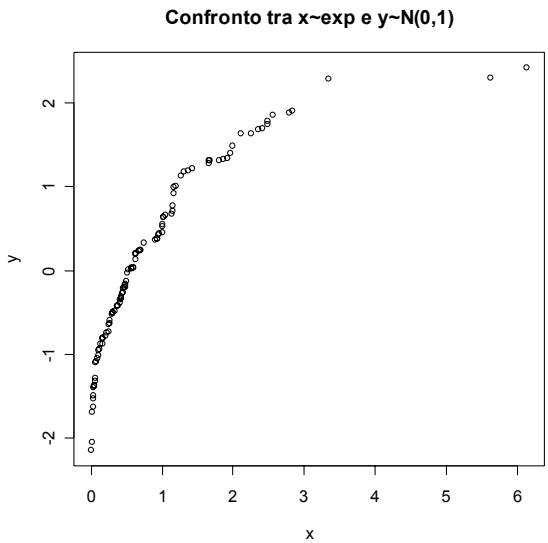
qqnorm: esempio




```
>  z<-rnorm(100);  qqnorm(z)
```

11

qqplot: esempio




```
> x <- rexp(100); y <- rnorm(100)
> qqplot(x,y, main="Confronto tra x~exp e y~N(0,1)")
```

12

Hist

hist genera istogrammi utilizzando un vettore numerico.

Esempi:

- hist(x):
  genera un istogramma utilizzando il vettore numerico x
- hist(x, nclass=n):
  genera un isotgramma con un numero n di classi
- hist(x, breaks=b, ...):
  i punti di break degli intervalli dei valori di x che delimitano le classi sono esplicitamente elencati con il parametro breaks
- hist(x, probability=TRUE)
  le colonne rappresentano frequenze relative invece che assolute

13

Visualizzazione di distribuzioni tramite istogrammi

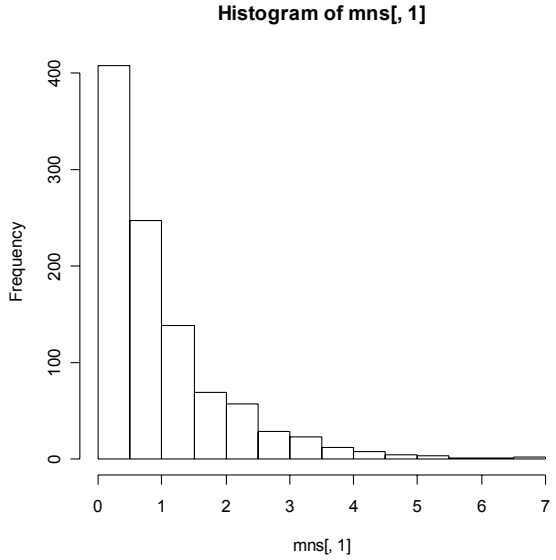
In accordo con il teorema del limite centrale si può far vedere come la distribuzione delle medie di campioni estratti da una distribuzione asimmetrica converga ad una normale, al crescere della cardinalità dei campioni:

```
> # generazione dell matrice dei dati estratti in accordo ad
> # una distribuzione esponenziale negativa
> data <- matrix( rexp( 1000 * 32), nrow = 32)
> mns <- cbind( data[ 1, ], # media dei campioni di 1
+ apply( data[ 1: 4, ], 2, mean), # media dei campioni di 4
+ apply( data[ 1: 32, ], 2, mean)) # media dei campioni di 32
```

Le distribuzioni possono essere visualizzate con istogrammi (vedi slide successive)

14

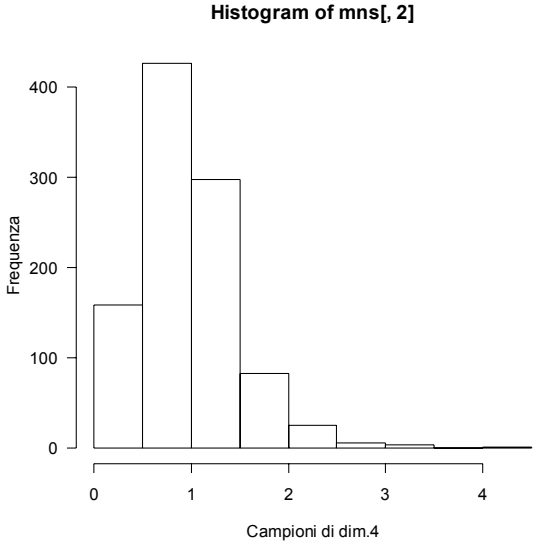
Istogramma delle distribuzione delle medie con n=1




```
> hist(mns[,1])
```

15

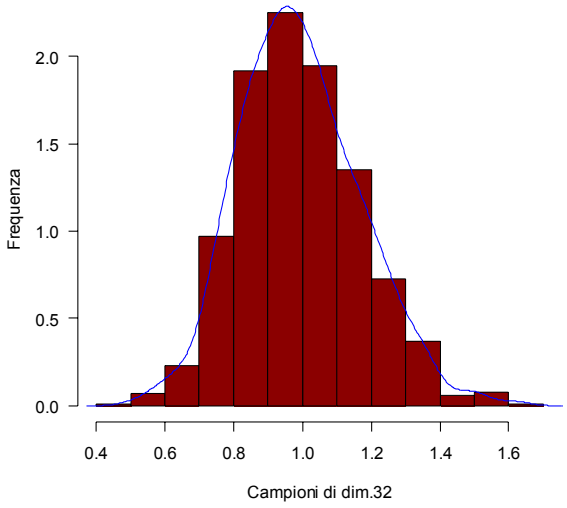
Istogramma delle distribuzione delle medie con n=4




```
> hist( mns[, 2], xlab = "Campioni di dim.4", las = 1)
```

16

Istogramma delle distribuzione delle medie con n=32

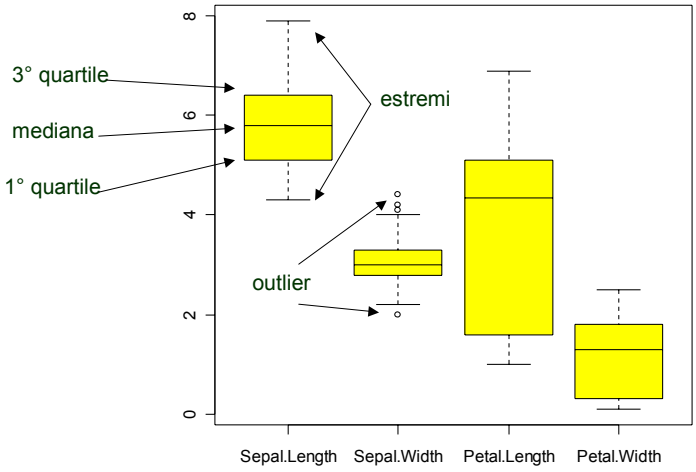



```
> hist( mns[, 3], main="", ylab = "Frequenza", xlab =
+ "Campioni di dim.32", las = 1, col = "darkred", freq=FALSE)
> lines(density( mns[, 3]), col = "blue")
```

17

Boxplot

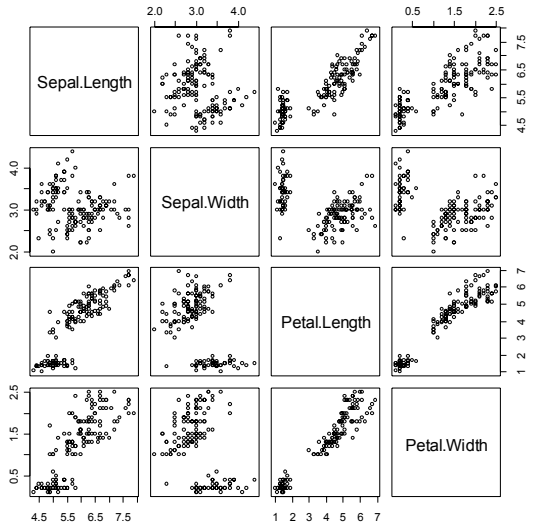
• Forniscono una descrizione grafica sintetica di un insieme di dati utilizzando semplici statistiche.




```
> data(iris); boxplot(iris[,1:4], col="yellow")
```

18

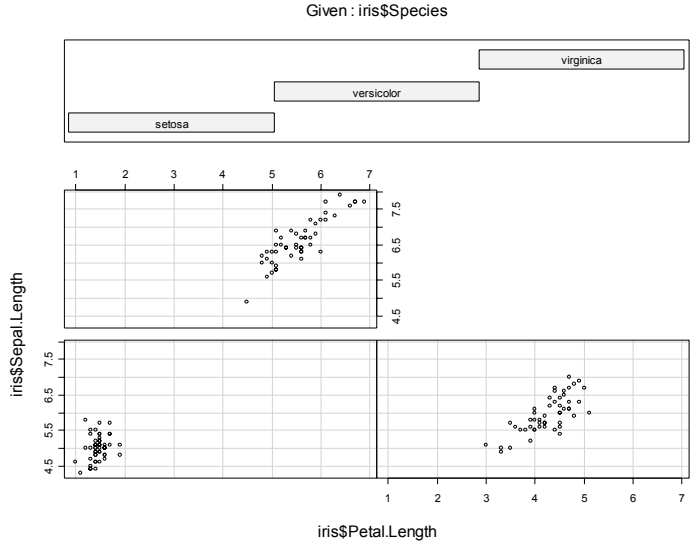
Rappresentazione di dati multivariati -1




```
> pairs(iris[,1:4])
```

19

Rappresentazione di dati multivariati -2




```
> coplot(iris$Sepal.Length ~ iris$Petal.Length | iris$Species)
```

20

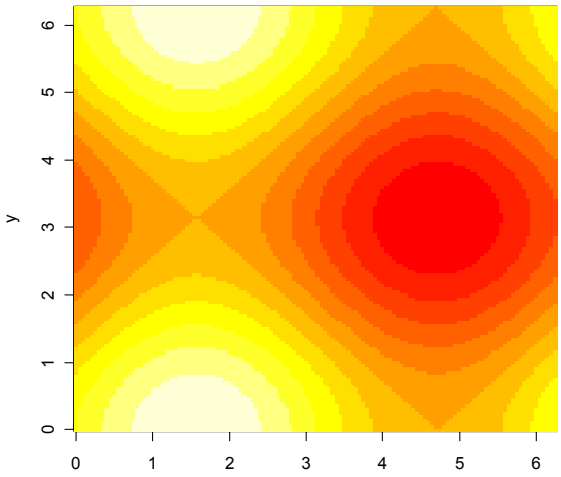
Funzioni per la grafica 3D

- image:
  permette di visualizzare grafici 3D, come immagini 2D, utilizzando diversi toni di colore per le altezze
- persp:
  permette di visualizzare superfici wireframe o a faccette piene
- contour:
  rappresenta una superficie 3D tramite curve di livello

  Ognuna di queste funzioni è dotata di diversi parametri che permettono diverse modalità di visualizzazione (v. help)

21

Image: esempio



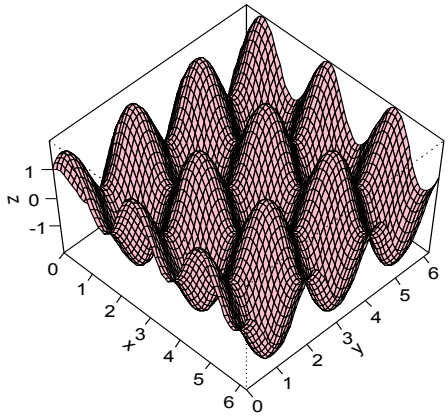
Rappresentazione della funzione:

f(x,y)=sin(x)+cos(y)

```
> x<-y<-seq(0,2*pi,by=0.05)
> z <- outer(sin(x),cos(y),"+") # crea la matrice z delle altezze
> image(x,y,z)
```

22

persp: esempio



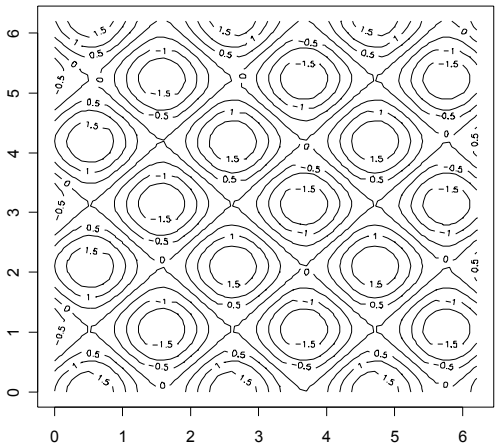
Rappresentazione della funzione:

f(x,y)=sin(3x)+cos(3y)

```
> x<-y<-seq(0,2*pi,by=0.1)
> z <- outer(sin(3*x),cos(3*y),"+") # crea la matrice z delle altezze
>  persp(x,y,z,phi=60,theta=45,d=10,col="pink",ticktype="detailed")
```

23

contour: esempio



Rappresentazione della funzione:

f(x,y)=sin(3x)+cos(3y)

```
> x<-y<-seq(0,2*pi,by=0.1)
> z <- outer(sin(3*x),cos(3*y),"+") # crea la matrice z delle altezze
>  contour(x,y,z)
```

24

Argomenti per le funzioni di alto livello

- Si possono passare diversi argomenti aggiuntivi alle funzioni di alto livello:

add=TRUE	Forza la funzione ad agire sul grafico corrente, aggiungendo nuove componenti al grafico
axes=FALSE	Sopprime la generazione automatica degli assi
log="x", log="y", log="xy"	Assi logaritmici
type=	Controlla il tipo di plot prodotto: type="p" plot per punti; type="l" plot di linee; type="b" plot di punti connessi da linee; type="h" plot di linne verticali dai punti all' asse. (v. help per descrizione completa delle opzioni disponibili)
xlab=stringa ylab=stringa	Etichette associate agli assi
main=string	Titolo del grafico
…	Esistono diversi altri argomenti specifici o meno per ogni funzione (v. help)

25

Funzioni grafiche di basso livello

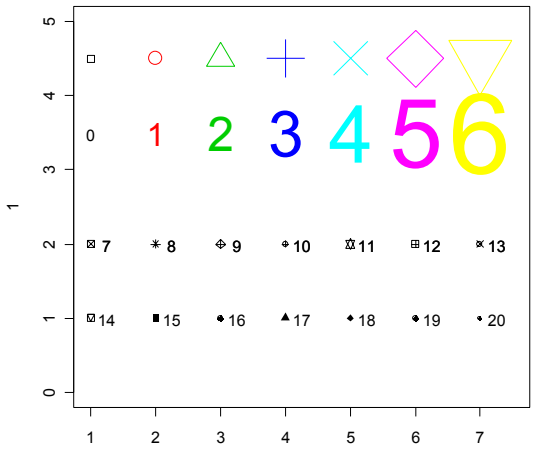
• E' possibile modificare un grafico generato con funzioni ad alto livello con funzioni di basso livello, che possono aggiungere ad es: punti, linee o testo ad un grafico esistente. Alcuni esempi (ma esistono molti altri comandi) sono:

points(x,y)	Aggiunge un punto al grafico corrente (in posizione x,y)
lines(x,y)	Aggiunge una linea al grafico corrente
text(x,y,label)	Aggiunge la stringa di testo label in posizione x,y
abline(a,b)	Aggiunge una linea di inclinazione a ed intercetta b
polygon(x,y,z, ...)	Disegna un poligono i cui vertici (ordinati) sono elencati come argomenti
legend(x,y,legend)	Aggiunge una legenda in posizione x,y
title(main.sub)	Aggiunge il titolo main ed opzionalmente un sottotitolo sub
axis(side,...)	Aggiunge gli assi nelle posizioni specificate da side

Le coordinate sono fornite in termini di *coordinate utente*, definite da precedenti comandi di alto livello

26

Esempio di funzioni grafiche di basso livello




```
> plot(1, 1, xlim=c(1, 7.5), ylim=c(0,5), type="n")
> points(1:7, rep(4.5, 7), cex=1:7, col=1:7, pch=0:6)
> text(1:7,rep(3.5, 7), labels=paste(0:6), cex=1:7, col=1:7)
> points(1:7,rep(2,7), pch=(0:6)+7)
> text((1:7)+0.25, rep(2,7), paste((0:6)+7))
> points(1:7,rep(1,7), pch=(0:6)+14)
> text((1:7)+0.25, rep(1,7),paste((0:6)+14))
```

27

Utilizzo dei parametri grafici

- Come abbiamo già visto è possibile modificare il comportamento delle funzioni grafiche
- In R è possibile modificare i parametri grafici secondo due modalità:
  1. Cambiamenti temporanei: cioè tramite il passaggio esplicito di argomenti alle funzioni grafiche (come già visto nelle slide preceenti)
  2. Cambiamenti permanenti: la funzione par permette di accedere e modificare permantemete i parametri del device grafico corrente, fino a che non viene nuovamente chiamata par.

28

Device driver

- R può generare grafici (a diversi livelli di qualità) per diversi display o dispositivi di stampa.
- Il ruolo dei device driver è di tradurre le istruzioni grafiche di R in una forma "comprensibile" per un particolare dispositivo di visualizzazione
- Per divenire attivo, ogni tipo di device driver dispone di una sua propria funzione di inizializzazione (vedi help("Devices") per una lista dei device driver disponibili).
- Successivamente l' output delle funzioni grafiche viene indirizzato al dispositivo selezionato.

29

Device driver: esempi

1. Apertura di una finestra in ambiente Windows (device aperto di default sul sistema utilizzato in laboratorio):

```
> windows() #
  l' output delle funzioni grafiche è indirizzato alla finestra
   corrente
> windows() # viene aperta un' altra finestra
  l' output delle funzioni grafiche è indirizzato alla nuova
   finestra
```

2. Creazione di un file grafico postscript

```
> postscript("grafico.ps") # apertura device postscript
> plot(y) # l'output grafico è indirizzato sul file ps
> dev.off() # chiusura del device postscript corrente
```

3. Creazione di un file grafico in formato jpeg

```
> jpeg("grafico.jpg")
> plot(y)
> dev.off()
```

30

Grafica interattiva

- Con la funzione locator è possibile sia ottenere le coordinate grafiche di un punto sul grafico, sia aggiungere un oggetto grafico in una posizione specifica tramite un click del mouse.
  Per esempio, si provino ad eseguire i seguenti comandi:

```
> y <-rnorm(100)
> plot(y)
> locator(5)
```

```
> y <-rnorm(100)
> plot(y)
> text(locator(1), "Punto critico")
```

- Con la funzione identify è possibile identificare punti particolari di un grafico tramite un click del mouse: per ogni click viene indicato l' indice del punto più vicino al click del mouse.
  Per esempio, si provino ad eseguire i seguenti comandi:

```
> y <-rnorm(100)
> plot(y)
> identify(y)
```

31

Grafica dinamica

- Attualmente R non ha funzioni grafiche built-in per supportare la grafica dinamica (ad es: rotazione di nuvole di punti, trasformazioni metriche o simili di grafici)
- Esiste comunque il *package R* xgobi che permette di accedere alle funzioni di grafica dinamica disponibili nel sistema Xgobi di Swayne, Cook e Buja: http://www.research.att.com/areas/stat/xgobi

32